160-TP-014-001

Leveraging Nominal Orbit Spatial Extent to Provide a Solution for the Archival of Geoscience Laser Altimeter System (GLAS) Products in ECS

Technical Paper

December 2001

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

David Heroux /s/	12/19/01
David Heroux, Science Office Engineer EOSDIS Core System Project	Date

SUBMITTED BY

Chuck Thomas for Robert Plante /s/	12/20/01
Robert Plante, Science Office Director EOSDIS Core System Project	Date

Raytheon Company
Upper Marlboro, Maryland

Abstract

Due to the irregular shape, and large extent, of spatial regions that often define swath-based products, a new feature has been introduced into the ECS System to capture these definitions. Referred to as NOSE, for nominal orbit spatial extent, this new archival feature accommodates the finer spatial extent of data products associated with a fixed, nominal set of orbits. The basis of this feature is founded on the characteristics of a known instrument (MISR). It was also designed to exploit the known constraints of the ECS implementation to provide a more efficient means of archiving and spatially searching on ECS products than would otherwise be possible, for products of this type, under the normal ECS archival mechanisms. Adapting this feature for use with similar products, like those currently being defined for the GLAS instrument, permits the re-use of ECS software that has already proven its worth in the field. Analysis of the available GLAS product requirements provide confidence that the NOSE solution is a viable one for the archival and retrieval of GLAS data in the ECS System. Additionally, results stemming from preliminary prototyping efforts, to spatially search on “GLAS-like” geographic extents (i.e., extremely narrow polygons), support this conclusion. Reasonably enough, some enhancement to the existing NOSE functionality has been identified to fully implement a solution in light of preliminary requirements expressed by the GLAS development team. The overall effort anticipated to achieve this solution appears reasonable, given the net savings in generated metadata, and the improvements in spatial search performance that can be expected with this approach.

Keywords: NOSE, GLAS, MISR, spatial search, archive, polygon

This page intentionally left blank.

Contents

3. The Application of NOSE: Impact of Implementation for GLAS Products

List of Figures

List of Tables

Appendix A: Toolkit Metadata Calling Sequence for Multi-value PSA

Appendix B: Toolkit Metadata Calling Sequence for Multiple Spatial Extents

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Purpose

Generally speaking, the archival of products within ECS occurs in a manner that is independent of the type of product. Such independence is made possible by the strict inclusion of core metadata within each product instance, or granule to use ECS parlance. Elements defined within this metadata provide the essential temporal and spatial components that ensure future retrieval of these archived products.

However, for performance and implementation related reasons, several constraints have been placed on the allowable spatial extent that can be defined for certain granules entering the archive. From early concepts of metadata and product requirements for the Geoscience Laser Altimeter System (GLAS) instrument, these restrictions would at first appear to disqualify GLAS products as potential ECS archive granules. But, as we have all come to learn, the general solution is not always the correct solution.

Indeed, this is the case for the archival of Multi-angle Imaging Spectro-Radiometer (MISR) instrument products. Late in the development of ECS, it was learned that the above mentioned restrictions would be exceeded by MISR. To accommodate this special case, a unique solution to the problem of archival was devised. Formally labeled Nominal Orbit Spatial Extent, or NOSE, this solution provides for the efficient archival and retrieval of MISR products without imposing an unreasonable resource load on the instrument development team. As with other ECS archival methods, the integrity of product search and order is maintained by the NOSE method.

In our quest to determine if the NOSE methodology is an appropriate one to use for the GLAS products, an analysis of the available archival and search requirements for those products must be conducted first. By then presenting the results of this preliminary analysis within the context of the NOSE implementation, and drawing on comparisons with the MISR case, we can determine if NOSE is a suitable solution for GLAS product archival.

As with most solutions, rarely is there a perfect fit; the NOSE method is no exception. Therefore, those GLAS issues that fall outside the scope of the NOSE implementation have been captured for subsequent resolution, but are not addressed in this paper. The degree to which GLAS product archiving is satisfied by the NOSE method may be influenced by these other factors.

1.2 Organization

This paper is organized as follows:

Section 1 of this paper defines the purpose and review policy for the capability defined herein.

Section 2 of this paper offers a detailed analysis of the spatial search mechanism within the ECS. This provides the context for reviewing the archiving requirements for the MISR/Terra instrument. From this discussion, the basis for the NOSE feature is discovered.

Section 3 of this paper leads into an analogous review of the conditions concerning a dissimilar instrument, GLAS, and the suitability of the NOSE method in meeting these conditions. The mechanics of applying the NOSE method are introduced here and assessed for this newer instrument. Several prototypes, generated as part of this assessment, are included here for completeness. A summary table of NOSE integration steps is also provided.

Throughout this paper, the reader should be aware of certain type conventions that were used to provide contextual aides. Where individual *italicized words* appear, the affected word(s) have connotations that are specific to the subject matter being discussed. Words in ***bold italic*** represent reserved words, or literal expressions; these are generally used to bring attention to attribute names that appear in ECS granule metadata. The use of **`courier bold`** text is used almost exclusively to represent values for ECS metadata, and the use of **times bold** is primarily to identify ECS system-defined names. Unless otherwise indicated (as is the case for some examples), these conventions apply to the entire document.

1.3 Review and Approval

This Technical Paper is an informal document approved at the Office Manager level. it does not require formal Government review or approval.

The concepts expressed in this Technical Paper are valid for the Release B system; extentions proposed here are anticipated to become part of the Release B system at at later data. Questions regarding technical information contained within this Paper should be addressed to the following ECS and/or GSFC contacts:

- o ECS Contacts
  - – David Heroux, Science Office Engineer, 301.925.073, dheroux@eos.hitc.com

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Company
1616 McCormick Drive
Upper Marlboro, MD 20785

1.4 References

333-CD-510 Release 5B SDP Toolkit Users Guide for the ECS Project

2. The Basis of NOSE:• Impact of MISR Science Requirements on the ECS Archival Subsystem

2.1 Overview

As with all new Earth observing instruments, each introduces a new set of goals and technologies that quite often translate into new requirements for the ground-processing segment of the mission. Although the ECS system was designed to address these requirements in a general-purpose way, it was not possible to account for every possible scenario in the initial design. Because of this, allowances have been made over time to support the unique requirements presented by some instruments. Many of these new requirements are addressed through the introduction of new *production rules* that govern the manner in which the ECS processing subsystem carries-out specific processing models, e.g., MODIS tile-based processing, MISR orbit-based processing. However, recent requirements have also emerged that have affected the archival aspect of the ECS system. So, by first presenting an overview of the fundamental archiving constraints that exist in the current ECS implementation, the impact of newer requirements on the ECS archiving subsystem, can be brought to light. With an understanding of these constraints, the approach taken to address some of these more recent requirements, can clearly be viewed as reasonable. It is then our goal to extend this approach to largely address the ostensible requirements offered by the GLAS instrument team.

2.2 ECS System Constraints[1]

The ECS archive chiefly serves as a repository both for data generated by EOS instruments, and for the higher-level products created by subsequent ground processing. This archive also serves as a vast source of data and products for the multi-disciplinary Earth science community. Adequately providing for the latter necessitates the use of search methods that are sensitive to both the temporal and spatial aspects of the data holdings. To efficiently accommodate user searches that are spatially-based, the ECS Science Data Server incorporates Autometric's Spatial Query Search (SQS™) engine[2]. However, as with all spatial search engines, there are inherent limits that need to be recognized. Accounting for these limits during the archival process affords the user the fullest range of functionality when conducting spatial searches for data. To impart the necessary level of understanding to would-be users of the ECS NOSE feature (i.e., data providers), some of the more significant limits are addressed here.

1 Much of the material presented in this section was graciously provided by Robert Hartranft, ECS Science Data Server Engineering Department, during impromptu discussions over the past year.

2 SQS is a trademark of Autometric Incorporated: www.autometric.com

2.2.1 Constraints on Spatial Query Server (SQS)

In recording the spatial extents of data, regions are often encoded through the use of minimum bounding rectangles (MBRs), or more complex polygonal shapes. Yet a fundamental constraint of SQS continues to be a limit on the maximum *angle* allowed for individual polygonal edges and for interior polygonal extents, i.e., geodetic angular extent of the longest interior arc. In SQS, the angle subtended by any single edge along a polygonal extent must be no greater than 180°; regional inclusion/exclusion rules within the spatial search engine become difficult to solve for very large angular extents[3].

Even for extents less than 180°, determining interior vs. exterior boundaries for large polygons sometimes prove challenging to SQS. To resolve this dilemma, Autometric introduced the *directed polygon*, where points bounding a polygon define an interior region when ordered clockwise, as viewed from above. To encode the intended order, the *point* data type in Sybase was extended to support a sequence number. To account for this in ECS metadata, the ***GringPoint*** object within the ***Gpolygon*** *container* was likewise enhanced to accommodate a sequence number.

Additionally, the floating-point accuracy of the Sybase *point* data type definition may lead to difficulties when archiving, or searching on, extremely narrow spatial extents. Empirical evidence shows there to be an effective resolution of 10m between any two points used to define a polygonal spatial extent. Points separated by a smaller distance may be indistinguishable from one another.

To support standard geographic searches, SQS defines an arc as 2 points connected by a great circle section, or geodesic. This rule applies to polygonal edges and is expressed during spatial searches against polygonal regions.

However, an anomaly can occur when very narrow polygonal extents are defined such that they contain both large angular arcs (> 90°) and smaller arcs (< 10°) whose end-points tend to fall in line with those belonging to the larger arcs; note that this is often the case for narrow orbital swaths. Such thinly stretched trapezoidal regions are subject to encountering this proximity restriction, where the geodesics that define the swath edges actually intercept, or cross over one-another. For obvious reasons, these shapes are forbidden by SQS and are therefore rejected during ECS archival attempts.

While each of these restrictions represents a hard constraint on the spatial extent allowed by SQS for archival purposes, other restrictions are not as decisive. This does not lessen their impact on the performance of the ECS archive however. To the degree possible, every effort must be extended to adhere to the common-sense restrictions presented in the sections that follow.

2.2.2 Quadtree Indexing in SQS

When granules are inserted into the ECS archive, the spatial regions that define the data are stored and indexed according to their geographical size and location. Under the existing ECS

[3] The v3.2.2 implementation of SQS cannot guarantee the outcome of search results for ***Gpolygons*** with > 180° arcs; this limitation no longer exists for bounding rectangles since the introduction of LLbox in this latest version.

implementation, indexing of spatial regions is accomplished by way of a *quadtree* structure that partitions the globe into East and West hemispheres of 16 *planes* each. The base *planes* (level 0) are reserved for those regions that exceed 90° in either longitude or latitude. Each successive plane (levels 1 – 15) divides it's parent into 4 *sub-planes* of equal angle and contain only those regions that are within ¼ the spatial extent of the parent *plane*. Naturally, all *global* products are indexed at the base level, while *non-global* products are indexed deeper within the quadtree structure. As a rule, a product is indexed to the smallest *plane* that completely contains its spatial region. It follows then that the smallest *planes* will contain the fewest regions and will therefore reference fewer products. The reader should note that *planes* within the *quadtree* structure, often referred to as nodes, are created as they are needed to store granule spatial extents and provide indexing for the actual granule.

When spatial searches are performed against a user-specified area, the result is a set of indexes where intersections are found. Since the intersection tests are performed directly against these *planar* extents, a subsequent look-up query is necessary to retrieve only those granule records with matching index values. Intuitively enough, fewer indexes translate into fewer matching granule records.

While this indexing scheme allows for more efficient searches of granules by spatial extent, <u>the gain is predominantly realized for granules with relatively small spatial extents; ideally about 1° in extent</u>[4]. The knife-edge is a balance between the number of *quadtree* nodes that must be searched initially, and the number of granule records that can be located at any particular index. Having too many granule records per node, translates into extensive database lookups. Having too many nodes, results in a geometrically increasing number of SQS intersection tests. Both factors are directly influenced by the size selected for the definition of spatial extents.

2.2.3 Spatial Search Performance in Science Data Server

Although spatial indexing is provided for by SQS, <u>the benefit of this is greatly diminished for spatial extents with more than a couple dozen vertices</u>. The reason is understandable. The more points required to define a spatial region in the database, the more complex the search space becomes. Multiply this by the number of potentially matching granules and you quickly get a sense of the magnitude of the problem.

On the surface, many of these constraints would appear to disqualify most swath-based ECS products, which by their very purpose are reasonably large in extent. To transcend these limitations, the NOSE solution, as derived in the following section, effectively permits granules to be defined through the use of multiple indexes. In point of fact, thoughtful application of this technique has been shown to alleviate most of the issues relating to the use of SQS, in handling large spatial extents within the ECS Science Data Server.

[4] Quadtree indexing performance is based on extrapolated Science Data Server projections of the times required to retrieve spatial information, and perform intersection tests using granules of various spatial extents.

2.3 Archiving Solution for MISR Global Products

2.3.1 MISR's Large Extents Stress SDSRV and Violate SQS Limits

The principal science requirement driving MISR pertains to the collection of aerosol measurements, sampled multiple times at small intervals, over the sun-lit portion of each orbit. In doing so, the instrument's array of cameras cut a swath along each orbit that is standardized to a fixed set of Landsat-based swaths, known as *paths*. In addition, each swath is partitioned according to a convention that identifies specific *scenes* referred to as *blocks*; in MISR's case there are 180 possible *blocks* for each *path.* The resultant geometry of the scene-based swath resembles a collection of stacked 2-dimensional blocks. The overall swath itself is rather large, extending more than 180° (approximately 1° per *block)* to provide terminator-to-terminator coverage year round. For any given orbit, the instrument footprint is comprised of a sequence of contiguous *blocks*, shifted in latitude with the seasons. Nominally this translates into 140+ *blocks* per granule. Using 4 vertices to define each *block*, this equates to roughly 560 points; discrete locations defined by latitude and longitude. Since MISR granules are orbit-based, their products tend to record the entire spatial extent of the daytime pass. If permitted to be archived in this state, each granule would easily exceed the previously identified, fundamental constraints of the ECS system. That is to say, too many points covering too large a spatial extent.

During an initial effort to reduce the number of points used in the definition of the MISR spatial extents, a proposal was made to record only a few points along the edge of the entire swath. Although this provided a reasonable facsimile of the MISR swath, and circumvented one of the archiving restrictions (i.e., too many points), the spatial extent itself was still too long. Having such a large extent effectively defines all such granules as *global* products. As we know from the previous section, all *global* products share the same indexing space within the archive database. The obvious drawback to this is that searches, subsequently performed for MISR products, could potentially involve every MISR granule ever archived! Predictably, granules archived in this manner will experience a decrease in search performance as a function of mission time.

2.3.2 High Resolution Requirement

Although the MISR *swath* is technically capable of being defined in terms of a single spatial extent for each *path*, other instrument team requirements, to allow for *block* based subsetting, preclude the use of *global* product attribution. Fortuitously, this same requirement effectively enforces the definition of granule spatial extent in terms of the smaller spatial regions referred to earlier as *blocks*.

Confronted with having to preserve the detailed spatial extent defined by MISR, a new mechanism was devised to overcome the inherent limitations that result from archiving *global* granules within ECS.

2.3.3 Taking Advantage of Repeating Swath of Instrument

In effect, this meant eliminating the *global* aspect of these granules. Logically, the only way to accomplish this was to effectively reduce the spatial extent of the granules. The challenge lay in accomplishing this while still permitting the original spatial extent to be searchable. Since the *quadtree*-indexing feature of SQS allows for small spatial regions to be stored and searched in an efficient manner, the answer was obvious. Archive each granule as though it were many small granules instead of a single larger one. The only detail that remained was to provide a mechanism for associating all of the smaller regions (i.e., the SQS indexes) with the actual granule.

Solving this detail was not as straight-forward. Normally, each time a granule is inserted into the ECS archive, a completely new definition for the granule*'s* geographic extent must also be communicated via metadata. Breaking this extent into many smaller regions would require a mechanism to associate each region with the granule upon insert. This would in-turn require either a series of individual granule insertions, one for each of the regions, or a single insertion with multiple granule association records to maintain a linkage to the original granule. Providing for this amount of detail in metadata could prove costly at runtime. Likewise, appending ever more granule association records to the archive database, with each successive insert, would become prohibitive in the long term.

The solution, as it turns out, was already provided by the MISR requirements themselves. Since MISR swaths are aligned to Landsat 7 *paths* (WRS specification for Landsat 4, 5, & 7[5]), and MISR has fixed sets of *blocks* established for each swath, and those Landsat 7 *paths* repeat, all of the essential granule association information is known at the start. This characteristic of the MISR mission serves to fix the number of granule association records in the database and establishes a predefined set of spatial indexes at the same time. All that is required by way of metadata is to convey the particular *path* number and the inclusive range of *blocks* imaged by the instrument for the current orbit. This negligible amount of metadata is sufficient to serve as a back-reference to an actual granule, should a spatial search ever intersect with the *block(s)* to which that granule is indexed.

The essence of this solution is that it works by associating fixed geographic regions with a set of fixed reference orbits (defined at ascending equator-crossing longitudes). In fact, the NOSE solution derives its name after the **s**patial **e**xtents that are predefined for a repeating set of **n**ominal **o**rbits, i.e., *block*-based *paths*.

2.3.4 Extension to EDG Client to Capitalize on NOSE Effort

To support the spatial subsetting requirement mentioned earlier, the EDG team has constructed a granule display client (CoverMap) that is sensitive to MISR spatial extents. In fact, the metadata attributes that record the indexing information, referred to in ECS as Product Specific Attributes (PSAs), became available to the EDG client via the V0 Gatewaywith ECS release 5B. This functionality allows the EDG client to display only the spatial extent supported by the granule*;* annotations for *block* and *path* number are supported as well.

2.4 Additional Background on the ECS NOSE Feature

Developed expressly for the purpose of archiving MISR products, this enhancement of the ECS production system effectively bypasses some inherent constraints of the core archival sub-system.

5 http://edcwww.cr.usgs.gov/glis/hyper/guide/wrs.html

Under the original system, swath-based products are constrained to use relatively few points for the definition of their spatial extents; it is typical for these products to have their swath extent defined as ***Gpolygon*** containers in metadata. Further restrictions are placed on the maximum angular extent of such products, e.g., cannot directly archive entire orbit swath products (unless the products are declared as *global*, which is not the desired intent here). For most EOS instruments, only a generalized regional extent is required to define a footprint for processing, e.g., minimum bounding rectangle (MBR). In such cases, granules can be readily inserted into the archive using standard rectangular-coordinate metadata to describe their extent. But for those EOS instruments that do not meet these criteria, an alternative mechanism for archiving products in ECS has been implemented - one that preserves the finer spatial requirement of the product.

2.5 Summary of NOSE Benefits and Restrictions

Principally, this method was designed to benefit those instruments that require a rather large spatial extent for the definition of their products, and where the use of *global* attribution is not appropriate for their purpose. Typically, such instruments are swath based. But this is not a strict requirement.

To support such instruments, this novel archiving mechanism exploits the repeating ground track pattern that is required for many EOS missions. However, for this same reason, this solution is only applicable to those instruments that can meet this requirement. While this solution circumvents the restrictions imposed by the base system, it does present some minor restrictions of it's own. For rationale related to the angular extent limitations in SQS, *block* extents should not exceed 10° in any one dimension. Also, for database performance reasons, the total number of blocks should be kept to as small a number as possible; certainly less than 1 million per instrument.

In return for investing in the up-front effort of predefining spatial extents for all possible orbits, only a minimal set of spatial metadata are required to be written with each granule. The spatial search performance that can be achieved by indexing granules to small *blocks* is unparalleled within the ECS. Finally, flexibility is achieved through support of additional orbit reference models, and the potential for higher resolution display of granule extents (EDG CoverMap)[6].

Although recently introduced, the NOSE mechanism has been deployed to all of the ECS archiving sites (DAACS). And with the release of version 3.0 of the EDG Client, the visual display of MISR spatial extents is now fully supported.

[6] The EDG Client display effort is handled through a separate contract from ECS.

3. The Application of NOSE: Impact of Implementation for GLAS Products

3.1 Overview

Apart from documenting the development of the NOSE feature, the primary purpose of this paper is to assess the suitability of the NOSE method for the archival of GLAS products. To some degree, this assessment relies on the identification of areas of commonality between the GLAS mission, and that of MISR, for which this method was originally intended. For those areas that are unique to the GLAS mission, adaptation of the NOSE feature must also be considered.

To start off, GLAS products are not technically considered to be swath products, since the instrument does not perform scanning observations. Nonetheless, the retrieved altimetry data are directly related to the geographic footprint of this nadir-pointing instrument. Although many GLAS products are believed to be discontinuous along the orbital *track*, due to varying surface properties, the spatial regions of data collection can be represented by multiple polygonal extents. For the purposes of satisfying ECS archival and search conditions, adopting a swath definition for GLAS products affords some advantage. By retrofitting GLAS products with swath-based attributes in metadata, establishing a set of synthetic swath extents, and facilitating the runtime determination of swath metadata within GLAS processing, these products will be able to transcend the inherent limitations of SQS to become bona fide, searchable granules within the ECS archive.

3.2 Evaluation of the GLAS Characteristics that Affect ECS Product Archival

As a polar-orbiting instrument like MISR, the GLAS instrument is designed to collect data along a highly inclined orbit and sample the entire globe many times over during its lifespan. While MISR collects reflected light from the atmosphere on the daylight side of the orbit, GLAS pulses a laser to measure the height of various ice-forms at strategic points along the entire orbit. Though similar to the MISR products in some respects, additional aspects of GLAS products are incomparable, and must be taken into account

3.2.1 Spatial Constraints Identified by the GLAS Development Team

While it is recognized that the GLAS team cannot identify any explicit requirements for spatial searches within the Earth science community, there remains an implicit requirement that all ECS archival products be searchable in this manner. This is in addition to similar requirements that address support for temporal searches against products. Since explicit geographical search requirements do not exist for GLAS products at the present time, a set of geographical constraints have been assembled here for the purpose of challenging the proposed implementation. These were generated from a basic understanding of the platform and instrument operations and granule production characteristics which are relevant to the subject matter at hand. Although these are not search requirements per se, they effectively constrain the spatial extent that can be defined for a granule and therefore indirectly influence the spatial requirements that may be generated for GLAS granules.

In order to support search requests from as wide a community as possible, it is assumed that searches will be performed on relatively localized regions, presumably nearest the poles. It is further assumed that searches will be conducted using the lat-lon box method (ECS normally translates this into LLBox searches in SQS).

To address the spatial search requirements that are recognized by the GLAS team, it is anticipated that all GLAS granules will contain additional metadata to define location in terms of GLAS-specific identifiers (***cycle, track, segment***). But without an established mechanism to index these attributes by their physical geographical extent, a V0 Client (e.g., EDG client) will be unable to support spatial searches in the fullest sense. For example, while specialized users will be able to perform searches based on track identifier[7], generalized users who rely on llbox methods for searching are effectively excluded from examining GLAS products.

Under general requirements, providing the elements necessary to support spatial search can be a complex undertaking. For GLAS products, this effort is exacerbated by several unique geographical constraints.

3.2.2 Small Instrument Footprint

In scanning the atmosphere for aerosol composition, the MISR instrument sweeps a large swath on the ground. Due to the function of the GLAS instrument, the equivalent ground swath is essentially defined by the nadir extent of the instrument's laser beam, which is nominally about 70m in diameter at the ground. Such a small cross-track dimension will require a precise definition for the spatial extent, in order to reduce the number of false returns during search attempts.

3.2.3 Partial and Multi-orbit Granule Production

While both instruments produce data for partial orbits, GLAS products have the potential to contain data that is non-continuous over multiple orbits, or even within a single orbit. Due to the multi-orbit aspect of GLAS daily granules, any spatial attribution of the data will involve a rather large angular extent, to say the least. This condition greatly exceeds the basic capabilities of SQS, unless of course the granules are defined as being of type *global*. Also, the potential for GLAS granules to be non-continuous implies the need for multiple definitions of spatial extent per granule; a feature that is currently not supported. Therefore ECS support for large, non-continuous segments, on a granule basis, will require enhancements to NOSE to accept multiple spatial index values from granule metadata.

[7] ECS Release 5B supports V0 Client search-by-PSA.

3.2.4 Imprecise Quarter-orbit Definition

To standardize the notion of partial orbit products as a science requirement, the GLAS instrument team (IT) has defined a *quarter-orbit* extent to be roughly equivalent to ¼ that of an entire orbit. For their purposes, these *segments* are partitioned by the 50° parallels, ± 1° owing to timing considerations. What at first appears to be a convenient boundary for the determination of large spatial *segments* within a single orbit, may not prove useful for the purpose of spatial partitioning through NOSE. Variabilities in the spatial definition need to be accounted for when defining spatial inclusion boundaries, e.g., locations within the latitude range [49°N .. 51°N] are indexed to the same spatial region (a.k.a. block).

3.2.5 Multiple Ground Repeat Tracks

As mentioned previously, the enforcement of a repeating reference orbit is a mandatory minimum requirement for NOSE support. The GLAS misssion however, supports multiple repeating reference orbits. The degree to which the different reference orbits overlap will determine how effective the spatial searches are at minimizing the number of false returns. In general, spatial extents should be designed to avoid or limit granule collisions where possible. So that, during search requests, granules are returned only for those *tracks* that intersect the search region. Reducing the frequency of false returns can be accomplished by defining the spatial extents as compactly as possible for each track, subject to the accuracy constraints cited earlier.

3.2.6 Poor Accuracy of Ground Repeat Track in Equatorial Regions

Lastly, in order to follow a designed reference orbit and maintain the accuracy of the ground *track* in the polar-regions, the platform has to perform a series of maneuvers. Unfortunately, as the platform drifts between these maneuvers, the accuracy of the ground *track* can degrade, most noticeably along the equatorial zones of the orbit. To compensate for any imprecision in the ground track, spatial extents may need to be expanded in these areas, again subject to the concerns of spatial overlap.

3.3 Additional Metadata Requirements

3.3.1 ECS Data Types

Without exception, all products require an Earth Science Data Type (ESDT) to be recognized as true granules in the ECS system. Each data type defines the entire set of metadata that is applicable to each granule of that type. Upon insertion into the ECS archive, a subset of the product's metadata is extracted for the creation of *Inventory* record(s). This *Inventory* forms the basis for subsequent search and order requests by the user community.

3.3.2 Collection Level Metadata : Attributes and DLL Modifications

Aside from the individual attributes that are typically used to define metadata on a per granule basis, there is another aspect of ESDT development; one that is normally transparent to the user. For every ESDT there exists a higher-order set of metadata that is used to describe entire collections of granules. Known as *Collection Level* metadata, these metadata are abstracted away from the individual granules, yet apply to each of them just the same. While most of the metadata at this level is created to document the common attributes of data for the user community, some are used internally by the ECS system to activate functionality that is data type specific.

Such is the case for data types that employ the NOSE feature. To tag these ESDTs, a special attribute named ***SpatialSearchType*** has been introduced. By setting a value of **`“orbit”`** for this attribute, the ECS archival sub-system will activate the NOSE-based insert functionality instead of using the standard spatial insert method.

Another modification that is necessary at the *Collection Level*, deals with the value assigned to the attribute ***DLLname***. Referred to as a dynamic link library, or DLL for short, this value identifies the compilation of shared objects that define the methods used to provide system services for ESDTs of a certain class. By default, the DLL assigned to most ESDTs provide for the basic granule-level services. For example, with most metadata, only single values are assigned to PGE-defined metadata attributes and therefore the DLL only provides for the creation of a single *Inventory* record for that attribute.

In order to impart the additional functionality to allow for attribute multiplicity (a requirement suggested in the previous section, and to be expressed in more detail in the section titled ‘***Writing New Metadata*’),** a measurable effort is required to upgrade the DLL referenced by the affected GLAS ESDTs. In essence, this will involve:

- cloning of the shared object that is referenced by the MISR ESDTs, **`libDsEsDtSyLgPoly.001sh.so`**, to create a GLAS specific version; a single version of the DLL should suffice for this purpose,
- parsing of the metadata containing multiple value fields and the creation of **`GlParameterList`** objects for each value,
- aggregation of the individual **`GlParameterList`** objects into a single list structure that satisfies the Science Data Server API for performing granule record updates to the **`DsMdOrbitCalculatedSpatial`** table.

While these modifications are straightforward, it is anticipated that several ESDTs will require the necessary modifications to support spatial indexing over the array of GLAS products.

3.3.3 Inventory Level Metadata : New PSA Definitions

In the spirit of reuse, the same database tables that are now used to store MISR spatial extents and granule-specific indexing records will also be used to record that same information for GLAS. Since these tables were created to address the MISR solution, several fields within these tables have names derived from the MISR PSAs used to collect their values. While it is recognized that there is not a direct relationship between *paths* in the MISR sense and *tracks* referenced by GLAS, their purpose is very similar. Also, while it is understood that MISR makes use of *block* numbers internally, a similar argument can be made for the use of *block* offset values to index granules for either instrument; in both cases, the term *block* refers to a polygon whose vertices have been pre-defined. Therefore, to preserve the intended use of these fields and ease the burden of maintainability, it is strongly suggested that the GLAS instrument team adopt

PSA naming conventions that mirror those used by the MISR IT for this purpose. The attributes used by MISR are given in Table 3-1, while the attributes recommended for GLAS are detailed in Table 3-2.

There are additional implications for the naming of these attributes, should the GLAS IT seek requirements for the display of GLAS granules by way of the ECS V0 Client[8]; the suggested names listed in Table 2 should lessen the impact here.

Table 3-1. PSAs used by MISR for NOSE Integration

PSA Name	Datatype	Max. Number of Values	Valids
SP_AM_PATH_NO	INTEGER:3	1	RANGE(1,233)
SP_AM_MISR_StartBlock	STRING:4	1	RANGE("001","180")
SP_AM_MISR_EndBlock	STRING:4	1	RANGE("001","180")

In establishing the limits for the parameters given in Table 3-2, an attempt was made to accommodate the nominal GLAS production scenario. Some assumptions were made about that production scenario, so the limits offered here may not reflect the actual GLAS case. Ultimately, the actual values used for these limits will need to be defined by the instrument team to better reflect the realistic scenario. Attention to the following details should facilitate that effort.

First, when defining a limit for the maximum number of attribute values, always consider the worst case scenario. A value of 15 was chosen here to account for a daily product that contains 1 contiguous range of polygons per *track*. In reality, there may be several non-contiguous ranges of polygons per track, so this limit may need to be increased to account for the number of discrete regions that are possible for any particular *track*.

Second, it is essential that the number of polygons per track be defined as a constant value, so that the range of *blocks* can be valid for all *tracks*. Although this is fixed, the actual extent of the polygons can vary as the requirements of a specific *track* dictate. The selection of 72, as the limit here, was based upon a nominal latitude extent of 5° per polygon. This yields about 200K unique polygons to cover all of the known GLAS tracks (~2842); this is in approximate agreement with the number of polygons defined for MISR instrument processing. While it was indicated earlier that a 1° extent was considered ideal from an SQS perspective, as a practical matter, this would involve the creation of too many polygons (~1,000K).

[8] In addressing MISR requirements, the EDG CoverMap application keys-off of these attributes in order to effect the display of granules in a meaningful way.

Table 3-2. Suggested PSAs for GLAS Integraton with NOSE

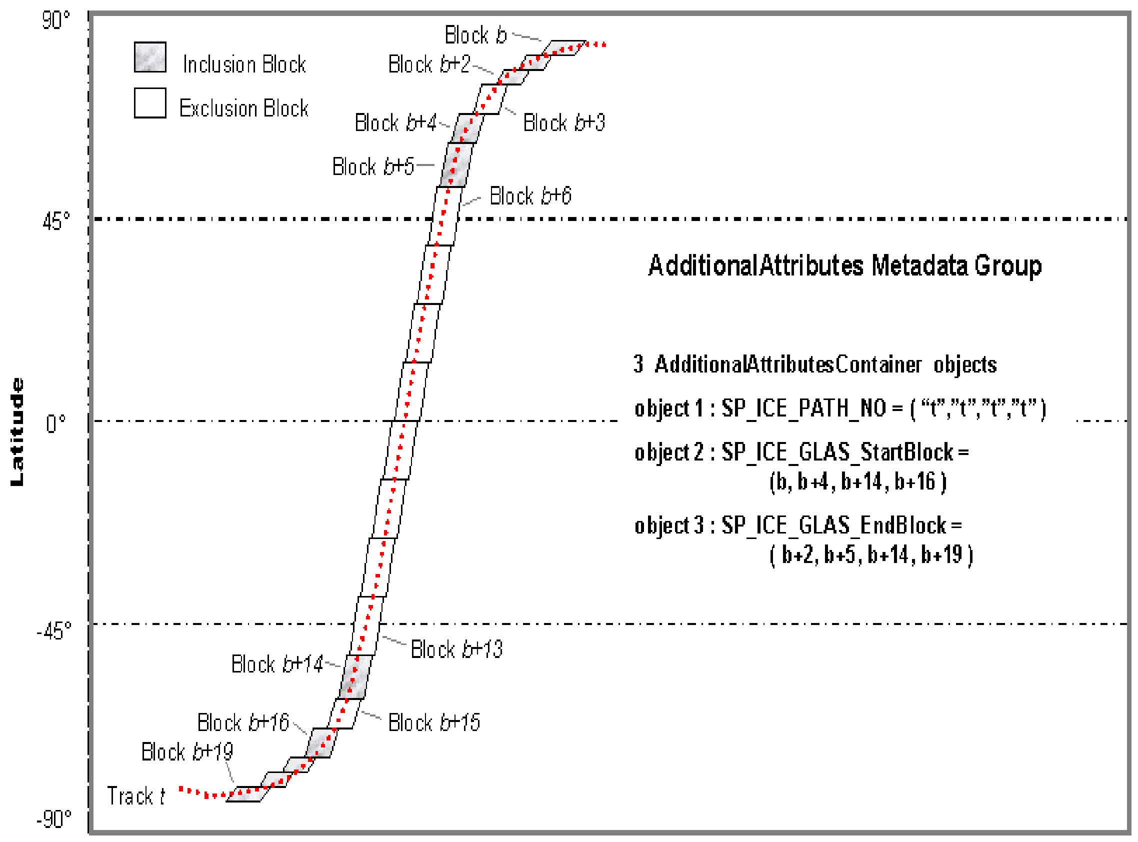
PSA Name	Datatype	Max. Number of Values	Valids
SP_ICE_PATH_NO	INTEGER:4	200	RANGE(1,2842)
SP_ICE_GLAS_StartBlock	STRING:3	200	RANGE("01","72")
SP_ICE_GLAS_EndBlock	STRING:3	200	RANGE("01","72")

3.3.4 Writing New Metadata

Though similar to their MISR counterpart in appearance, the new set of GLAS PSAs will need to differ in one important respect. When these attributes are being written to metadata, each will consist of one or more values. The actual number of values used will depend on two factors: one being the number of non-contiguous regions being recorded for a particular orbit; and two, the number of orbits (ground *tracks*) that are represented by the granule. As noted previously, GLAS granules will range in temporal extent anywhere from a partial orbit to several orbits. Whereas MISR granules are confined to a single orbit (*WRS path*) so their ESDTs are defined to only support single values per PSA.

The mechanics of this are straightforward. Rather than setting attributes through a single parameter = "value" pair, an entry will consist of a list of values as in parameter = (value1, … valueN). To allow for this multiplicity on attribute values, an additional modification to the *Inventory Level* of affected ESDTs is required. Given that each PSA object, by default, contains a ***Num_Val*** attribute set to **"1"**, this must be replaced by the attribute ***MaxOccurrences***, with appropriate limits as suggested in Table 2 under the heading "*Max. Number of Values*".

Figure 3-1 depicts a sample mapping of multiple spatial extents to *Inventory* level PSAs. Sample code, furnished in Appendix A, reveals the Toolkit metadata calling sequences that are used to write multiple values to a single PSA. As just mentioned, this attribute must be configured to accept multiple values.



90°
45°
0°
-45°
-90°
Latitude
Inclusion Block
Exclusion Block
Block b
Block b+2
Block b+3
Block b+4
Block b+5
Block b+6
Block b+13
Block b+14
Block b+15
Block b+16
Block b+19
Track t
AdditionalAttributes Metadata Group
3 AdditionalAttributesContainer objects
object 1 : SP_ICE_PATH_NO = ("t","t","t","t")
object 2 : SP_ICE_GLAS_StartBlock = (b, b+4, b+14, b+16)
object 3 : SP_ICE_GLAS_EndBlock = (b+2, b+5, b+14, b+19)

Figure 3-1. Mapping Multiple Spatial Extents to Additional Attributes Metadata for a Partial Orbit (required)

3.3.5 Providing Gring Values

Taken together, all of the preceding measures to augment the standard metadata description of data, complete the ECS system requirements for ESDTs to become NOSE compliant. Beyond this, however, there are two additional requirements that relate to the production aspect of NOSE.

First, there is a requirement for the IT to provide a static table of spatial extents by *track number*. This data is used to activate the NOSE feature and must be bulk-loaded into the Science Data Server database during initialization of the ECS application server **`EcDsStArchiveServer`**. The initialization script **`DsDbSqsBcpOrbitPolygon`** already exists to perform this task on MISR data; adaptation to support GLAS data is not expected to have a significant impact on ECS.

Second, there is the standard metadata requirement for the IT to furnish the actual spatial content within the product metadata itself. Ordinarily, this spatial information, defined by ***GRingPoint*** objects in metadata, is used directly in the creation of *Inventory* records for products. Unlike common spatial products however, NOSE-based products encode this spatial information as a single string. Within the *Inventory*, it is maintained solely for retrieval purposes during product search and order requests, i.e., it is not used to fulfill spatial search requests. Satisfying this requirement implies a need for GLAS production to determine and/or access pre-defined spatial extents for each *track* and supply, in metadata, that portion of each *track* that is represented by the product.

Technically, there is a requirement for all ECS archival products to have their spatial content embedded within metadata. However, unless there is a science team requirement to provide actual spatial metadata with their products, this last requirement can be waived; the prior requirement fully satisfies the ECS criteria for spatial search via the ECS V0 Client. **Please note that this exception only applies to products that are "NOSE-enabled".**

If spatial content is to be written to metadata, then care must be exercised to account for the surface extent represented by the granule. For example, unlike the MISR products, which represent the entire extent of aerosol data gathered during the daytime portion of the orbit[9], many GLAS products are based on surface classification, which typically represent a subset of the total data gathered over an entire orbit. This science aspect of the GLAS mission has ramifications for the expression of spatial content in metadata, which should only contain a record of the extents that coincide with data represented by the product. To account for this in metadata, a series of ***GpolygonContainer*** objects could be written to define the entire possible spatial extent, less the individual exclusion extents where no relevant data are represented. Figure 3-2 depicts a sample mapping of staggered spatial extents to *Inventory* metadata. Translation of this illustration into actual Toolkit metadata calls can be achieved by following the example calling sequences provided in Appendix B.

[9] Individual cameras have staggered activation and deactivation times at terminator crossings due to the variable viewing geometry of the instrument.

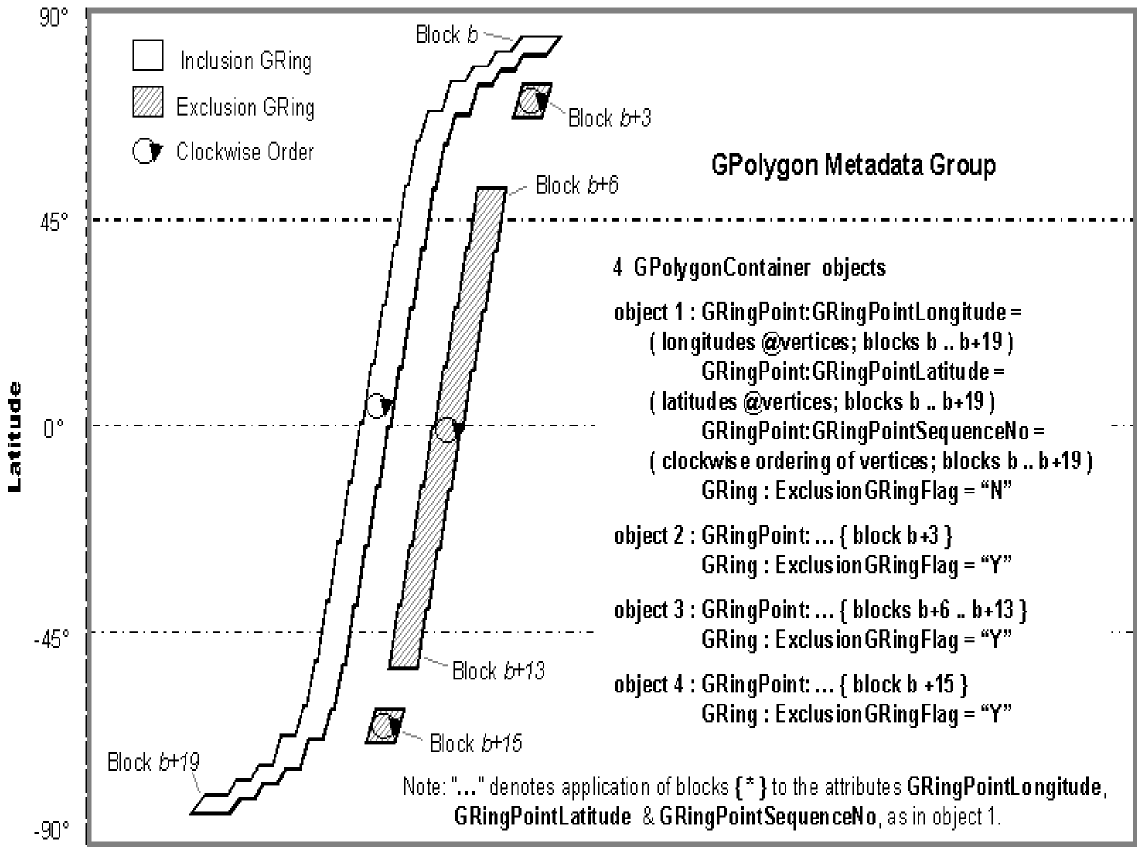

90°
45°
0°
-45°
-90°
Latitude
Inclusion GRing
Exclusion GRing
Clockwise Order
Block b
Block b+3
Block b+6
Block b+13
Block b+15
Block b+19
GPolygon Metadata Group
4 GPolygonContainer objects
object 1 : GRingPoint:GRingPointLongitude =
(longitudes @vertices; blocks b .. b+19)
GRingPoint:GRingPointLatitude =
(latitudes @vertices; blocks b .. b+19)
GRingPoint:GRingPointSequenceNo =
(clockwise ordering of vertices; blocks b .. b+19)
GRing : ExclusionGRingFlag = "N"
object 2 : GRingPoint: ... { block b+3 }
GRing : ExclusionGRingFlag = "Y"
object 3 : GRingPoint: ... { blocks b+6 .. b+13 }
GRing : ExclusionGRingFlag = "Y"
object 4 : GRingPoint: ... { block b +15 }
GRing : ExclusionGRingFlag = "Y"
Note: "..." denotes application of blocks { * } to the attributes GRingPointLongitude, GRingPointLatitude & GRingPointSequenceNo, as in object 1.

Figure 3-2. Mapping Multiple Spatial Extents to Gpolygon Metadata for a Partial Orbit (optional)

3.4 Defining GLAS Spatial Extents

Having suggested three new GLAS PSAs in Table 2, exactly what values does one assign to them? The answer may not be obvious at this point. Though, it is likely that these metadata values will be derived from the temporal boundary conditions that take effect during the production of GLAS granules. Determination of this temporal range often resolves the spatial extent of generated products; this is especially true for nadir pointing instruments, like GLAS. While the responsibility for providing PSA and ***GRing*** metadata rests with the instrument development team, the basis for their determination is not left to the instrument team to define alone. To this end, the following section will explore various factors that should be considered when defining the geometry of GLAS spatial polygons. Practical guidance for the determination of PSA values will be offered under the section heading '***Polygon definitions for track segments'***. In doing so, the finer details of establishing a predefined set of spatial extents, including formatting them for compatibility with ECS, will be laid-out along with a suggested strategy for accessing spatial content during GLAS production.

3.4.1 Synopsis of Early Prototyping Efforts

Early recognition of the possible use of NOSE for application to GLAS products led to the development of a small prototyping effort to explore this idea. The preliminary goals of this effort were twofold: first to determine if portions of the GLAS ground *track* could be approximated by a simple polygon, and second, to verify that such a polygon could satisfy the basic search requirements of the ECS system. Having partially achieved these objectives, the prototyping team[10] refined their goals to investigate methods for reducing the deviation between their experimental model and that of an actual reference ground *track.* A brief description of these efforts, along with a summary of the team's results, is provided here for completeness and to provide a partial basis for the conclusion of this paper.

For the initial prototype, a theoretical ground *track* was devised to form a basis for experimentation. Apart from providing a visual testament to the accuracy of an artificial spatial approximation, i.e., a polygon, this track also serves as a datum about which any approximation can be measured. In defining a spatial extent to enclose this *track*, 2 polygons were constructed with vertices near 50°N ascending, 50°N descending, and 50°S descending. Parallels of 50° were selected as end points for the polygons, since these locations are known to have some relevance to GLAS production. The spacing of end points at these locations was set to 3.6 seconds (0.0010°), or the width of the nominal 70 m ground *track* at 50°. The points defining these polygons were subsequently loaded into the database (Sybase) to generate regions to search against. A reasonable depiction of this case is displayed in Figure 3-3.

Please note that the polygons described above are not accurately represented in this figure. In actuality, the diagram is a display of 2 continuous great circle arcs that are slightly phase shifted to either side of the central arc representing the *track*. As both arcs describing the polygons extend beyond 180°, they cross over each other, as can be observed for the edges of polygon 1. This is an unwanted artifact that can be attributed to the technique used to create the display. In reality, the edges of such a polygon are not allowed to intersect (a condition not tolerated by SQS). To ensure this for the prototype, the actual polygons were individually defined with a common set of end points where they met. Note that although the adjoining edges of the 2 polygons no longer form a continuous great circle arc (as depicted in the diagram), the individual edges of each polygon are in fact treated as great circle arcs by SQS during spatial tests for intersection.

[10] Dr. Siri Joda Khalsa led the prototyping effort wih Dr. Xin-Min Hua conducting the effort and performing analysis of the results; Figures 3-3, 3-4 & 3-5 and Table 3-3 were annotated and reproduced here with their permission.

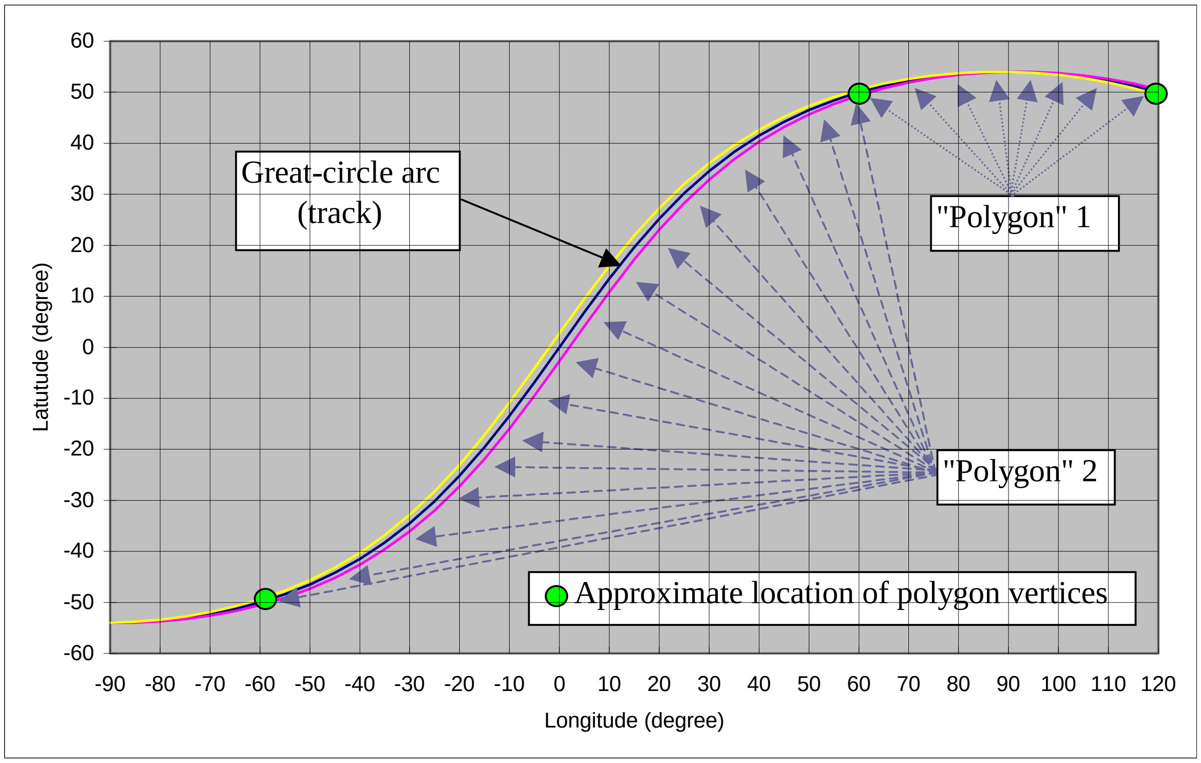


Figure 3-3. Simulated Orbit Approximated by Sections Divided at latitudes 50°S and 50°N

With the prototypical spatial polygons registered in the database, a series of searches were performed to determine if intersections with the polygons could be achieved. In each case, this was accomplished using lat-lon boxes to define the search area of interest; the use of lat-lon boxes is the same methodology employed by ECS (SQS LLbox) when performing search and order requests via the V0 Gateway. In conducting these searches, an initial set of search boxes were defined so as to flank the spatial polygons, not to intersect them. Then by incrementally increasing the size of the search boxes in the latitudinal, or longitudinal direction, until a search result could be achieved, the team was able to approximate the resolution at which intersections occurred with the spatial polygons. As a result of this exercise, the spatial polygons were determined to be very thin along the entire length of the enclosed ground track. This exercise revealed a minimum deviation of 0.0001° at 90°E for the first polygon, maintaining a constant deviation of 0.0005° for the second polygon.

In a refinement of this first prototype, a section of the GLAS mission reference orbit was used to represent the spacecraft sub-satellite point for a single orbit. Since the instrument nadir footprint approximates that of the ground *track*, this provides a suitable model for prototyping the spatial extent of a GLAS product. Essentially, this next prototype was modeled by a set of great circle arcs that best approached the ground *track* along specific portions of the orbit. For instrument team purposes, GLAS products are partitioned into 4 sections per orbit; aptly named *quarter-orbit* products. To follow-suit, the prototype was divided into 4 sections per orbit, though only 2 sections, representing half an orbit, were used for analysis. <u>It is important to note here that while complete polygons were not generated as part of this effort, the lengths of great circle arc, within the sections of interest, can be visualized as the *edges* of polygons for all intents and purposes.</u>

Ideally, any polygon defined for this prototype should barely enclose the nominal 70 meter cross-track footprint of the GLAS instrument. Realistically, this is not achievable due to fundamental constraints imposed by SQS and the laws of orbital mechanics. For any arc, defined by 2 connecting points, SQS interprets the arc as being defined by great circle geometry. However, the ground *track* of the GLAS instrument does not precisely follow a great circle due to precession of the spacecraft relative to the Earth's rotation. Though beneficial to the instrument, this *track* has unfortunate consequences for the prototype. Chiefly, requiring any polygon to be sufficiently wide enough to completely envelop a non-great circle *track*. Given these constraints, a polygon, defined by 2 points at the ends of a *quarter-orbit track segment*, will diverge from the *track* most notably along the polar-regions of the *track segment*. Note that the deviation here will be significantly greater than that achieved in the first prototype, which used a great circle arc in modeling the *track* itself. Though still a good approximation, the deviations between the sample *track segment* and any long polygon may be too great to uniquely define spatial extents for all of the required *quarter-orbit segments*.

In defining the *quarter-orbit* partitions, the GLAS instrument team had indicated a natural boundary of ±50° latitude. This was also the boundary selected for the second prototype; A graph of which is shown in Figure 3-4. As demonstrated by the position of the polygon *edges*, with respect to the reference *track*, the use of only 4 points to define a suitable *quarter-orbit* spatial extent does not appear to be sufficient; at least not in the polar latitudes.

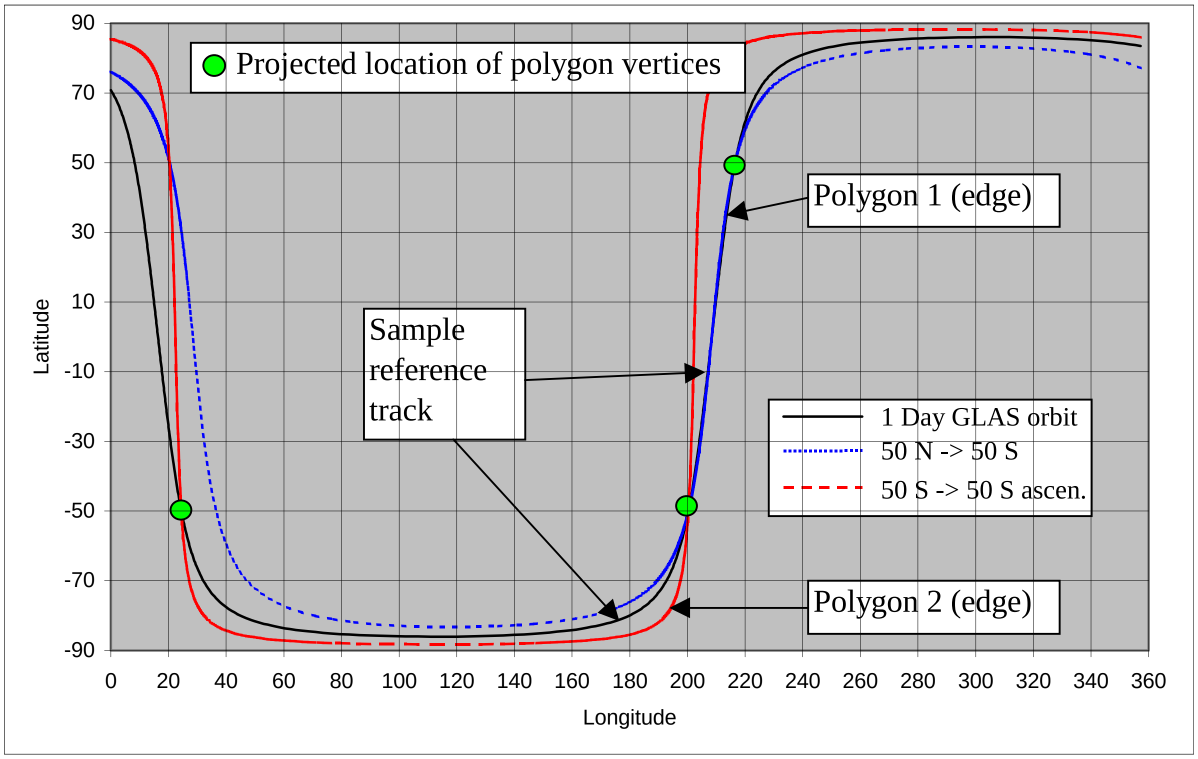


Figure 3-4. ICESAT Orbit Approximated by Sections Divided at latitudes 50°S and 50°N

In an effort to reduce the large deviations near the poles and provide a better approximation to the sample ground *track*, a third prototype was defined with segment boundaries at ±60° latitude. This last attempt clearly shows, in Figure 3-5, that the *track* deviations are significantly reduced along the polar section of the prototype polygon *edge*. However, even with this improvement, the deviations introduced by long polygons are still too great to accommodate the large number of GLAS spatial extents with any degree of acceptable accuracy.

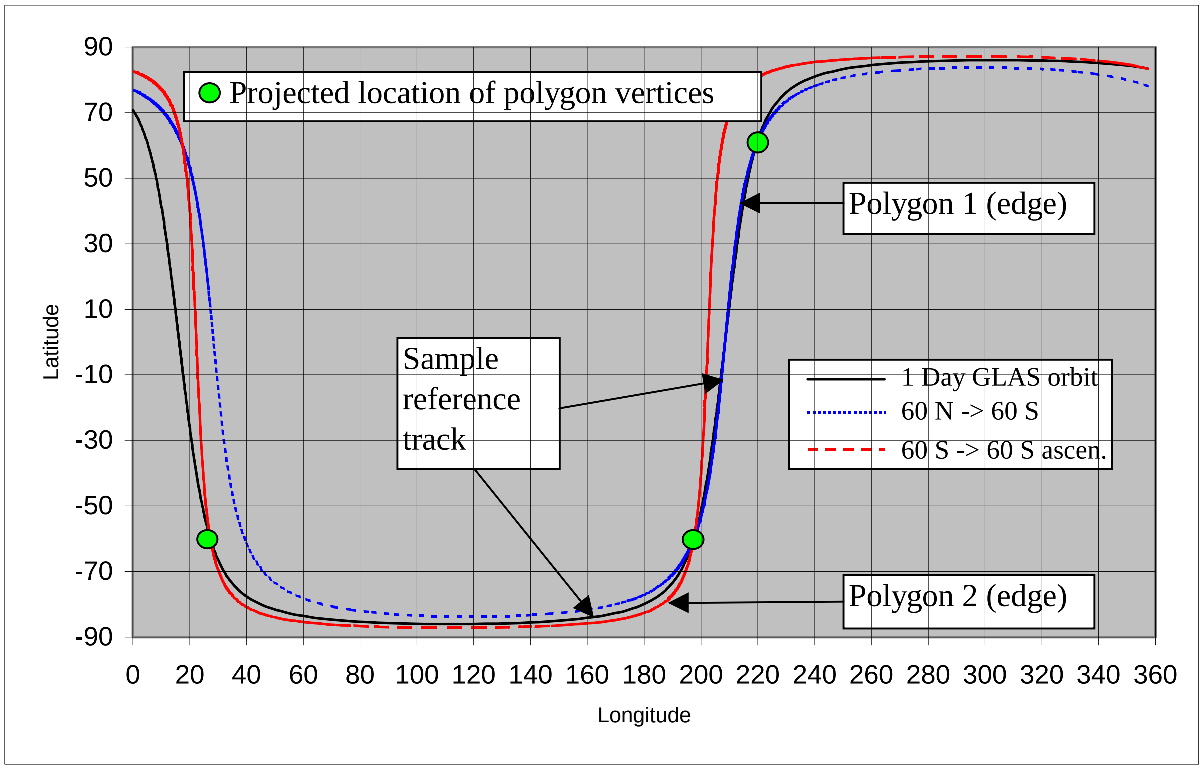


Figure 3-5. ICESAT Orbit Approximated by Sections Divided at latitudes 60°S and 60°N

For comparison purposes, Table 3-3 lists the maximum latitudinal deviations for the prototype *edges* against the sample ground *track* for 2 *quarter-orbit* sections divided at ±50° latitude and ±60° latitude respectively.

Table 3-3. Maximum Ground Track Deviation for Great Circle Polygons at Quarter-orbit Divisions

Track division	**50°N / 50°S**		**60°N / 60°S**	
¼ Orbit section ι	**50°N → 50°S**	**50°S → 50°S (a)**	**60°N → 60°S**	**60°S → 60°S (a)**
Max. deviation	2.6°	11.0°	4.6°	3.8°

ι All latitudes represent the descending node unless specified as ascending (a).

3.4.2 Conclusions Drawn from Prototypes

From these efforts, we may conclude the following:

a. Although polygon edges of long angular extent (e.g., 50°N-50°N) do approximate the ground track (which is not along a great circle arc, due to precession), polygon edges of shorter extent (e.g., 60°N-60°N) more closely approximate the ground track in polar latitudes. The assertion being made is that the shorter the angular extent of the polygon edge, the better it will approximate the ground track. For angular extents of 10° latitude, the deviation from the track center is expected to be negligible.

b. Since a single great circle arc diverges significantly from the spacecraft track in the polar-regions, a continuous "polyarc" defined by many smaller adjoining polygon edges can be used to more precisely enclose the ground track in these regions.

c. Even though the continuous extent outlined by a collection of polygons, suitably designed to enclose the track, will no longer contain a single great circle arc, the individual polygons themselves will continue to be treated as a collection of great circle arc edges by SQS. The significance here is subtle but critical just the same. It would be erroneous to design a polygon with 4 vertices to cover a large, yet narrow, angular extent, unless that extent confined a great circle arc. Any other arc will likely contain points that fall outside of the polygonal boundary, making it impossible for SQS to locate the external arc sections. Since this restriction does not extent to polygonal regions of small angular extent, to the same degree, it is practical to define the natural instrument spatial extent, or footprint, as a series of smaller polygons.

d. Since the spacecraft ground track more closely coincides with that of a great circle arc near the equatorial regions, larger polygons may be used to define portions of the extent in these latitudes. Noting however, that the spacecraft sub-satellite point may be off by as much as 1km from the nominal ground track in these regions[11], it may be prudent to extend the width of the polygonal segments here. Also, to the degree that this deviation is quantifiable, many smaller segments of varying width may be used to account for any variability in the spacecraft track with respect to latitude.

e. It is possible to use LLbox search techniques to locate granules defined by narrow polygonal extents.

[11] Estimate attributed to spacecraft drift offered by Anita Brenner (Raytheon ITSS) during GLAS CDR.

3.4.3 Repeat Ground Track Definitions : 183-day & 8-day

It was stated previously that to be compliant with the NOSE implementation, a spacecraft/instrument combination must employ a repeating ground *track*. Unlike its Terra-MISR counterpart however, the ICESat-GLAS platform introduces the additional variability of having multiple repeating ground *tracks*. On the surface, this condition would seem to violate the implied constraint. However, by defining a unique set of ground *tracks* across all repeat cycles, the thrust of the requirement can be satisfied.

It is understood from a GLAS document[12] that only 2 repeat cycles are currently defined. One identifies a 183-day cycle for nominal mission operations, while the other describes an 8-day cycle for calibration and validation. It is also known that fixed reference orbits are used to define each repeat cycle. Therefore, if the maximum number of *tracks* can be computed for each cycle (presumably from the number of orbital passes in a 24-hour period), then the combined *track* count could be used as an upper limit for the unique *track* number. Also note that the individual sets of *track* numbers are not required to be contiguous, merely unique in a repeatable way. By convention, the same *track* number always begins at a specific equator crossing longitude for a particular reference orbit; typically this is the ascending node crossing. If the ground *track* for one repeat cycle were coincident with another (i.e., same sub-satellite path), then it may be advantageous to reuse the appropriate *track* number. This may not be the case for GLAS however.

Table 3-4 lists the suggested ranges for each GLAS repeat cycle. If necessary, additional reference orbits can be accommodated, subject to the ECS conditions required for each. It should also be noted that there is an upper limit of 32K on the number of unique *track* numbers that can be defined for any single instrument.

Table 3-4. Suggested List of unique Ground Track Numbers

Reference Orbit	Number of Tracks	Unique Track No.
Mission : 183 day cycle	2723	1 .. 2723
Cal/Val : 8 day cycle	119	3001 .. 3119
Other : N day cycle	0	4000 .. ?

[12] Document *Requirements.doc* furnished by the GLAS development team.

3.4.4 Polygon Definitions for Track Segments

Assessing the earlier section on '***Providing Gring values'***, where the tasks of defining GLAS spatial polygons and incorporating these within the product metadata are spelled out, this would appear to require a substantial amount effort. While the generation of spatial polygons, and the modification of metadata *write* software are both one-time efforts, the actual writing of metadata occurs for each GLAS production run and may impact runtime performance to a measurable degree.

The actual generation of GLAS spatial polygons will itself involve the determination of 4 points for each polygon. With each point being defined by its latitude and longitude position, and a sequence number that defines its order within the clockwise collection of points that comprise the polygon. While the number of spatial polygons must be constant for all *tracks*, the actual number will be determined by the GLAS IT, having given thoughtful consideration to factors such as angular extent, surface classification containment resolution, and sheer quantity of polygons per mission. Likewise, the shape of each spatial polygon will be determined by several factors, including: polar-region sensitivity, and *track* deviation with latitude, which may be *track* specific. Another important factor to consider during this process is the spatial width of the polygons. Where it is practical to do so, all polygons should be as narrow as possible to reduce the number of overlapping regions. Consideration for the latter will help reduce, or eliminate, the number of false returns during product search and order requests.

Since the total number of spatial polygons that need to be generated is quite large (conceivably greater than 100K), it is anticipated that the IT will automate this effort through development of a polygon generation tool. To facilitate the subsequent effort of loading the polygon data into the ECS database, it would be ideal if such a tool generated its output in a format that can be directly applied to the ECS effort. Such a format is provided in Figure 3-6. If this is not possible, then the use of an intermediate format will be required.

```
A|1|track_1|block_1
B|pt1_lat|pt1_lon|0
B|pt2_lat|pt2_lon|0
B|pt3_lat|pt3_lon|0
B|pt4_lat|pt4_lon|2

A|1|track_1|block_2
          .
          .
          .
```

- Items in **bold** are fixed for ECS purposes
- Longitude (lon) values are expressed in decimal degrees (4 radix places), with values in the range [-180.0000° .. 0 .. +180.0000°] [+E]
- Latitude (lat) values are expressed in decimal degrees (4 radix places), with values in the range [-90.0000° .. 0 .. +90.0000°] [+N]
- Repeat **A,B,B,B,B** records for all blocks per track, and all tracks

Figure 3-6. Example of Sample Format of Spatial Polygons Required for the ECS Initialization of NOSE[13]

Upon completion of the above task, the final (optional) effort will be to adopt, or develop, a mechanism for accessing these data at runtime in order to facilitate the writing of ***GRing*** metadata. For any particular granule however, only a subset of these spatial polygons will need to be written, i.e., those for which the granule includes data along the spatially enclosed *track(s)*. One method used to accomplish this, and in fact the method employed by the MISR IT, is to define a series of *track (path)* specific ancillary files that contain the associated ***GRing*** information.

As an aside, the ECS Production sub-system takes advantage of the PSA values embedded within certain ancillary granules, to retrieve those that match the *track* values required at runtime. For example, in MISR production, which uses the Orbit-based processing rule, the current orbit is translated into a WRS *Path*, the value of which is then used in a query to retrieve ancillary granules that are required for that *path*.

3.5 Summary of Required Tasks

In an effort to condense the material just presented, Table 3-5 is offered here, which summarizes the various efforts that are necessary for the GLAS IT to take advantage of the NOSE feature. It is hoped that this table will provide the reader with a quick reference of the separate efforts required, along with the relative level of effort anticipated for each.

[13] Format required by EDG is (track, block, POLYGON((pt1_lat,pt1_lon), (pt2_lat,pt2_lon), (pt3_lat,pt3_lon), (pt4_lat,pt4_lon))) …

Table 3-5. Summary of Efforts Required for GLAS Integration with NOSE

Task	**Task Description**	**Relative Level of Effort**	**Principal**
1	Add ***SpatialSearchType*** to Collection Level of each GLAS ESDTs	small	ECS
2	Define 3 new PSAs for each ESDT to record swath extent of granule; establish attribute limits (see Table 2)	small	GLAS
3	Add new PSAs to existing GLAS ESDTs	small	ECS
4	Set MaxOccurrences in PSAs	small	ECS
5	Define *track* numbers for each reference orbit (see Table 4)	small	GLAS
6	Define all polygons (*blocks*) for each track; points ordered in the clockwise sense. (estimate of effort based on the creation of 200,000 blocks)	considerable	GLAS
7	Tabularize points by *track* and *block* (ideally using format shown in Figure 3-6)	moderate	GLAS
8	Create new DLL for GLAS ESDTs to account for multiple PSA values.	moderate	ECS
9	Update Science Data Server Initialization Script	small	ECS
10	Upgrade SIPS Processing to supply *track*(s)	N/A	GLAS
11	Upgrade SIPS Processing to supply polygons for each processed *track*	N/A	GLAS
12	Modify module(s) to retrieve *track*(s) & start/end *block*(s); write metadata for PSAs (see Appendix A)	N/A	GLAS
13	**(optional)** Modify module(s) to retrieve polygon data; write GRing metadata for polygons (see Appendix B)	N/A	GLAS

3.6 Outstanding ECS Archival an Search Issues for GLAS Products

While most GLAS data correspond to the nadir track of the instrument, there are thought to be some cases where this rule is not upheld. One such case pertains to the "off-pointing" of the instrument to make observations, at locations that are not coincident with the nadir position. Unless such locations fall within the range of the spatial extent defined for local sections along that *track*, the metadata used by NOSE to index this data will be incorrect.

If, however, a set of "off-pointing" locations were predefined for each *track*, then it is conceivable that the spatial extents could be expanded along the appropriate sections; though doing so may lead to overlap with the spatial extents of neighboring sections assigned to different *tracks*. If this is not possible, and these "off-pointing" locations can only be determined dynamically, then some alternate mechanism for defining spatial extent would appear to be warranted for these cases.

3.7 Conclusion

It is generally understood that all products entering the ECS archive must be capable of supporting spatial, and/or temporal search requests, lest they be archived for subscription purposes only. This precondition suggests that all products necessarily contain metadata that adequately describe both the temporal and spatial extent of the associated data. However, there are some restrictions within the system that constrain the methods used to define this metadata. Some of these constraints are necessary to ensure reasonable performance during spatial searches. Others are mandated to avoid common search restrictions. A definition of spatial metadata that observes these restrictions will endure for the life of the ECS archive.

Unearthing the content of this metadata, for the GLAS mission, necessitated a rather deep excavation into the accessible spatial requirements of that mission. To reveal how such metadata might be formed, a study of the approach used for another instrument, with similar spatial requirements, was performed. To this end, the NOSE solution, as applied to MISR instrument granules, was presented. Along the way, a reasonable amount of detail was imparted to document this feature, and to provide sufficient depth for studying its potential impact on GLAS. Where the GLAS instrument departs from the norm (general case being application to MISR), an analysis of the additional impact, levied by the apparent GLAS spatial constraints, was offered to fortify the case for leveraging NOSE. Independently, the conclusions reached for the GLAS prototyping effort serve to illuminate a justification of this approach.

While some issues remain unresolved, it is clear that the NOSE approach offers a solution for the archival of most GLAS products under ECS. The extent of the effort required to implement this solution for GLAS, may be significant, given the potentially large volume of spatial elements anticipated for the GLAS mission.

Abbreviations and Acronyms

CDR	Critical Design Review
DAAC	Data Active Archive Center
DLL	Dynamic Link Library
ECS	EOSDIS Core System
EDG	EOS Data Gateway
ESDT	Earth Science Data Type
GLAS	Geoscience Laser Altimeter System
IT	Instrument Team
LLbox	Latitude-Longitude box (A.K.A. lat-lon box)
MISR	Multi-angle Imaging Spectro-Radiometer
NOSE	Nominal Orbit Spatial Extent
ODL	Object Description Language
PSA	Product Specific Attribute
SDSRV	Science Data Server Subsystem
SIPS	Science Investigator-lead Processing System
SQS™	Spatial Query Server (Autometric Incorporated)
WRS	World Referencing System

This page intentionally left blank.

Appendix A: Toolkit Metadata Calling Sequence for Multi-value PSA

The mechanism used to provide multiple index values is the ECS Metadata Toolkit library (MET). The calling sequence used to assign multiple values is similar to that used to assign singular values. The difference is that the former requires the creation and assignment of a partial array.

Detailed information and examples on the use of the Toolkit Metadata routines can be located in the SDP Toolkit User's Guide for the ECS Project {333-CD-510-002}. This document can be retrieved from the ECS Data Handling System at http://edhs1.gsfc.nasa.gov/.

This is a sample block of 'C' code that uses ECS Toolkit MET routines (**highlighted**) to assign NOSE attributes for 2 extents along the same track:

(**disclaimer:** this code is provided for illustrative purposes only and is therefore not supported as an actual unit of software.)

```
#include <PGS_PC.h>
#include <PGS_MET.h>
#include <HdfEosDef.h>
#define INVENTORYMETADATA 1
/*
For the Toolkit Metadata routines to generate output properly,
-- entries similar to the following must be defined in the
-- process control file (PCF)
-- #Input Section
-- 400|GLAS_Product_ESDT.MCF|/SIPS_directory||||1
--
-- #Output Section
-- 40|GLAS_Product_File.HDF|/SIPS_directory||||1
-- 50|GLAS_Product_File.Binary|/SIPS_directory||||1
--
-- #Runtime Parameters Section
-- 1000||50:1
--
--
-- NOTE1: LID=66,666 file name, and directory are arbitrary
-- NOTE2: Toolkit reserves LIDs [10000 .. 10999] for its own use.
*/

#define GLAS_PRODUCT_MCF_LID  400
#define GLAS_HDF_PRODUCT_LID  40
#define GLAS_BIN_PRODUCT_LID  1000  /* Not 50 as one might expect –
                                       see sample PCF above */
#define ODL_IN_MEMORY         0
#define MAX_NOSE_VALS         200
#define REF_ORBIT_TRACKS      2723
#define TRACKS_PER_GRANULE    14
#define BLOCKS_PER_TRACK      15
#define MAX_GRANULE_BLOCKS    BLOCKS_PER_TRACK * TRACKS_PER_GRANULE
```

```
int main()

       char                    productPath[PGSd_PC_FILE_PATH_MAX];
       PGSt_integer            version = 1;

       extern AGGREGATE        PGSg_MET_MasterNode;
       int32                   sdid;
       char*                   datetime = NULL;
       PGSt_integer            tracks[MAX_NOSE_VALS];
       PGSt_integer            startBlks[MAX_NOSE_VALS];
       PGSt_integer            stopBlks[MAX_NOSE_VALS];
       char                    no_exclusion_flag[2] = {'N','\0'};
       char                    yes_exclusion_flag[2] = {'Y','\0'};
       char                    trackNumbers[MAX_NOSE_VALS][5];
       char                    startBlksLst[MAX_NOSE_VALS][5];
       char                    stopBlksLst[MAX_NOSE_VALS][5];
       char*                   StartPtr;
       char*                   StopPtr;
       char*                   trackPtr;
       int                     tracks[MAX_NOSE_VALS];
       char                    track_name[30] = "SP_ICE_PATH_NO";
       char                    start_block_name[30] = "SP_ICE_GLAS_StartBlock";
       char                    end_block_name[30] = "SP_ICE_GLAS_EndBlock";
       PGSt_integer            metFileId = GLAS_PRODUCT_MCF_LID;
       PGSt_MET_all_handles    handles;
       PGSt_MET_handle         handle; /* Handle for the GPolygon
                                          attribute section. */

       PGSt_SMF_status         ret = PGS_S_SUCCESS;
            .
            .
            .
```

```
/*
--  Retrieve the name * directory location of product file (from PCF)
*/
  version = 1;
  ret = PGS_PC_GetReference(GLAS_HDF_PRODUCT_LID, &version, productPath);
            .
            .
            .
/*
--  Capture metadata values (insert your methods here)
*/
  datetime = (char *) malloc(30);
  strcpy(datetime, "2000-04-11T12:30:45.7Z");

  /* Set values for 2 block ranges; 1 per segment; terminate partial array */
  startBlks[] = {1, 3, INT_MAX};
  stopBlks[] = {2, 4, INT_MAX};

  /* Set values for 2 track segments; same track in this example */
  tracks[] = {1, 1, INT_MAX};
  for (s=0;s<2;s++)
  {
    sprintf(trackNumbers[s][],"%04d",tracks[s]);
    sprintf(startBlksLst[s][],"%04d",startBlks[s]);
    sprintf(stopBlksLst[s][],"%04d",stopBlks[s]);
  }
  trackNumbers[s] = NULL;
  startBlksLst[s] = NULL;
  stopBlksLst[s] = NULL;
  StartPtr = startBlksLst;
  StopPtr = stopBlksLst;
  trackptr = trackNumbers;

  /* Set clockwise series of points for GRing information;
     the ordering of block vertices is determined entirely
     by the sequence number assigned to each vertex; as a
     result, individual blocks will not necessarily have
     sequentially numbered vertices!                      */


      PGSt_integer         sequences[MAX_GRANULE_BLOCKS*4] =
                                    {12,11,2,1,10,9,4,3,8,7,6,5,INT_MAX};
      PGSt_double          latitudes[MAX_GRANULE_BLOCKS*4] =
                                    {14.962560,14.962560,
                                      9.977400, 9.977400,
                                      9.977400, 9.977400,
                                      4.991090, 4.991090,
                                      4.991090, 4.991090,
                                      0.004747, 0.004747,
                                                 DBL_MAX};
```

```
       PGSt_double          longitudes[MAX_GRANULE_BLOCKS*4] =
                                      {-46.036600,-46.054000,
                                       -45.356300,-45.338600,
                                       -45.338600,-45.356300,
                                       -44.669600,-44.651600,
                                       -44.651600,-44.669600,
                                       -43.988000,-43.970000,
                                                     DBL_MAX};

       PGSt_integer         masked_seqs[MAX_GRANULE_BLOCKS*4] =
                                       {4,3,2,1,INT_MAX};

          /* sequences 10,9,4,3 from above */
       PGSt_double          masked_lats[MAX_GRANULE_BLOCKS*4] =
                                       {9.977400, 9.977400,
                                        4.991090, 4.991090,
                                                   DBL_MAX};
       PGSt_double          masked_lons[MAX_GRANULE_BLOCKS*4] =
                                       {-45.338600,-45.356300,
                                         44.669600,-44.651600,
                                                      DBL_MAX};
/*
--  Initialize MCF for setting metadata
*/
  ret= PGS_MET_Init(metFileId, handles);
  if(ret != PGS_S_SUCCESS)
  {
    fprintf(stderr,"initialization failed\n");
    exit( 0 );
  }

/*
--  Create/Open HDF Product file to write metadata to HDF file
*/
  sdid = SWopen((char *)productPath,DFACC_RDWR);
  sdid = SDstart(productPath, DFACC_CREATE);
  if(sdid == FAIL)
  {
    fprintf(stderr,"SDstart failed\n");
    exit( 1 );
  }


/*
--  Set values for Inventory metadata
*/
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA], "RangeBeginningDateTime",
&datetime);

/*
--  Set values for NOSE attribution
*/
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
"AdditionalAttributeName.1", &track_name);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
"AdditionalAttributeName.2", &start_block_name);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],
"AdditionalAttributeName.3", &end_block_name);
```

```
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA], "ParameterValue.1",
&trackPtr);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA], "ParameterValue.2",
&startPtr);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA], "ParameterValue.3",
&stopPtr);

/*
--  Defining Spatial Coverage for granule
*/

/* PLACEHOLDER – SEE APPENDIX B */


--  Writing metadata to HDF file
*/
  ret = PGS_MET_Write(handles[INVENTORYMETADATA], "metadata", sdid);
  if(ret != PGS_S_SUCCESS && ret != PGSMET_W_METADATA_NOT_SET)
  {
    if(ret == PGSMET_E_MAND_NOT_SET)
    fprintf(stderr,"some mandatory parameters were not set\n");
  else
    fprintf(stderr,"HDF Write failed\n");
    }

/*
--  Writing metadata to ASCII file "GLAS_Product_File.Binary.met"
--  This file is generated automatically based on the name of the
--  indirectly-referenced Product file
*/
  ret = PGS_MET_Write(handles[ODL_IN_MEMORY], NULL,
(PGSt_integer)GLAS_BIN_PRODUCT_LID);
  if(ret != PGS_S_SUCCESS && ret != PGSMET_W_METADATA_NOT_SET)
  {
    if(ret == PGSMET_E_MAND_NOT_SET)
      fprintf(stderr,"some mandatory parameters were not set\n");
    else
      fprintf(stderr,"ASCII Write failed\n");
  }

            .
            .
            .
} /* end main */
```

Output in the granule associated metadata file (*.met) should resemble the following for NOSE attribution ...

```
GROUP = INVENTORYMETADATA
            .
            .
            .
GROUP                  = ADDITIONALATTRIBUTES
    OBJECT                 = ADDITIONALATTRIBUTESCONTAINER
      CLASS                = "1"

      OBJECT                 = ADDITIONALATTRIBUTENAME
        CLASS                = "1"
        NUM_VAL              = 1
        VALUE                = "SP_ICE_PATH_NO"
      END_OBJECT             = ADDITIONALATTRIBUTENAME

      GROUP                  = INFORMATIONCONTENT
        CLASS                = "1"

        OBJECT                 = PARAMETERVALUE
          NUM_VAL              = 2
          CLASS                = "1"
          VALUE                = ("0001","0001" )
        END_OBJECT             = PARAMETERVALUE

      END_GROUP              = INFORMATIONCONTENT

    END_OBJECT             = ADDITIONALATTRIBUTESCONTAINER

OBJECT                 = ADDITIONALATTRIBUTESCONTAINER
      CLASS                = "2"

      OBJECT                 = ADDITIONALATTRIBUTENAME
        CLASS                = "2"
        NUM_VAL              = 2
        VALUE                = "SP_ICE_GLAS_StartBlock"
      END_OBJECT             = ADDITIONALATTRIBUTENAME



      GROUP                  = INFORMATIONCONTENT
        CLASS                = "2"

        OBJECT                 = PARAMETERVALUE
          NUM_VAL              = 2
          CLASS                = "2"
          VALUE                = ("1","3")
        END_OBJECT             = PARAMETERVALUE

      END_GROUP              = INFORMATIONCONTENT

   END_OBJECT             = ADDITIONALATTRIBUTESCONTAINER

OBJECT                 = ADDITIONALATTRIBUTESCONTAINER
      CLASS                = "3"
      OBJECT                  = ADDITIONALATTRIBUTENAME
        CLASS                 = "3"
        NUM_VAL               = 1
        VALUE                 = "SP_ICE_GLAS_EndBlock"
      END_OBJECT              = ADDITIONALATTRIBUTENAME

      GROUP                   = INFORMATIONCONTENT
      CLASS                 = "3"

        OBJECT                  = PARAMETERVALUE
          NUM_VAL               = 2
          CLASS                 = "3"
          VALUE                 = ("2","4")
        END_OBJECT              = PARAMETERVALUE

      END_GROUP               = INFORMATIONCONTENT

    END_OBJECT              = ADDITIONALATTRIBUTESCONTAINER

  END_GROUP               = ADDITIONALATTRIBUTES
            .
            .
            .
END_GROUP = INVENTORYMETADATA
END
```

This page intentionally left blank.

Appendix B: Toolkit Metadata Calling Sequence for Multiple Spatial Extents

Providing this metadata can be achieved through clever use of the same Toolkit Metadata calling sequences that appear in Appendix A. Here, multiple “GRing” objects are used to define entire orbital spatial extents from which partial extents are masked-out. The result is a set of actual spatial extents for a granule. The “GRing” objects are comprised of various GRing attributes, namely: “GRingPointLongitude”, “GRingPointLatitude”, GRingPointSequenceNo”, and “ExclusionGRingFlag”. The first 2 attributes define sets of geodetic points. The second attribute establishes the ordering of those points. The last attribute determines if the spatial extent, defined by the first 3 attributes, is to be included, or excluded from the definition of this granule.

Insertion of the following ECS Toolkit MET routines, into the sample block of ‘C’ code in Appendix A, assigns GRing attributes for 2 extents, along the same track, by excluding an intermediate extent that lies between the 2:

```
/*
--  Set Total Spatial Coverage Possible for granule
*/
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"GRingPointSequenceNo.1",
sequences);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"GRingPointLongitude.1",
longitudes);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"GRingPointLatitude.1",
latitudes);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"ExclusionGRingFlag.1",
&no_exclusion_flag);

/*
--  Mask out spatial regions that are not represented by the granule
*/
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"GRingPointSequenceNo.2",
masked_seqs);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"GRingPointLongitude.2",
masked_lons);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"GRingPointLatitude.2",
masked_lats);
  ret = PGS_MET_SetAttr(handles[INVENTORYMETADATA],"ExclusionGRingFlag.2",
&yes_exclusion_flag);
```

Output in the granule associated metadata file (*.met) should resemble the following for GRing attribution ...

```
GROUP = INVENTORYMETADATA
      .
      .
      .
 GROUP                  = SPATIALDOMAINCONTAINER
GROUP                  = HORIZONTALSPATIALDOMAINCONTAINER
  GROUP                  = GPOLYGON

      OBJECT                 = GPOLYGONCONTAINER
        CLASS                = "1"
        GROUP                  = GRINGPOINT
          CLASS                = "1"
          OBJECT                 = GRINGPOINTLONGITUDE
            NUM_VAL              = 12
            CLASS                = "1"
            VALUE                = (-46.036600,-46.054000,-45.356300,-
                                     45.338600,
                                    -45.338600,-45.356300,-44.669600,-
                                     44.651600,
                                    -44.651600,-44.669600,-43.988000,-
                                     43.970000)
          END_OBJECT             = GRINGPOINTLONGITUDE

          OBJECT                 = GRINGPOINTLATITUDE
            NUM_VAL              = 12
            CLASS                = "1"
            VALUE                = (14.962560,14.962560, 9.977400, 9.977400,
                                     9.977400, 9.977400, 4.991090, 4.991090,
                                     4.991090, 4.991090, 0.004747, 0.004747)
          END_OBJECT             = GRINGPOINTLATITUDE

          OBJECT                 = GRINGPOINTSEQUENCENO
            NUM_VAL              = 12
            CLASS                = "1"
            VALUE                = (12,11,2,1,10,9,4,3,8,7,6,5)
          END_OBJECT             = GRINGPOINTSEQUENCENO
       END_GROUP               = GRINGPOINT

       GROUP                   = GRING
          CLASS                 = "1"
          OBJECT                 = EXCLUSIONGRINGFLAG
            NUM_VAL              = 1
            CLASS                = "1"
            VALUE                = "N"
          END_OBJECT             = EXCLUSIONGRINGFLAG
        END_GROUP              = GRING
      END_OBJECT             = GPOLYGONCONTAINER

      OBJECT                 = GPOLYGONCONTAINER
        CLASS                = "2"
        GROUP                  = GRINGPOINT
          CLASS                = "2"
          OBJECT                 = GRINGPOINTLONGITUDE
            NUM_VAL              = 4
            CLASS                = "2"
            VALUE                =  ( -45.338600,-45.356300,-44.669600,-
                                       44.651600 )
          END_OBJECT             = GRINGPOINTLONGITUDE

          OBJECT                 = GRINGPOINTLATITUDE
            NUM_VAL              = 4
            CLASS                = "2"
            VALUE                =  ( 9.977400,9.977400,4.991090,4.991090 )
          END_OBJECT             = GRINGPOINTLATITUDE
```

```
            OBJECT                 = GRINGPOINTSEQUENCENO
              NUM_VAL              = 4
              CLASS                = "2"
              VALUE                = (4,3,2,1)
            END_OBJECT             = GRINGPOINTSEQUENCENO
         END_GROUP              = GRINGPOINT

         GROUP                  = GRING
            CLASS                = "2"
            OBJECT                 = EXCLUSIONGRINGFLAG
              NUM_VAL              = 1
              CLASS                = "2"
              VALUE                = "Y"
            END_OBJECT             = EXCLUSIONGRINGFLAG
          END_GROUP              = GRING
        END_OBJECT             = GPOLYGONCONTAINER

    END_GROUP              = GPOLYGON
  END_GROUP              = HORIZONTALSPATIALDOMAINCONTAINER
END_GROUP              = SPATIALDOMAINCONTAINER
      .
      .
      .
END_GROUP = INVENTORYMETADATA
END
```

This page intentionally left blank.